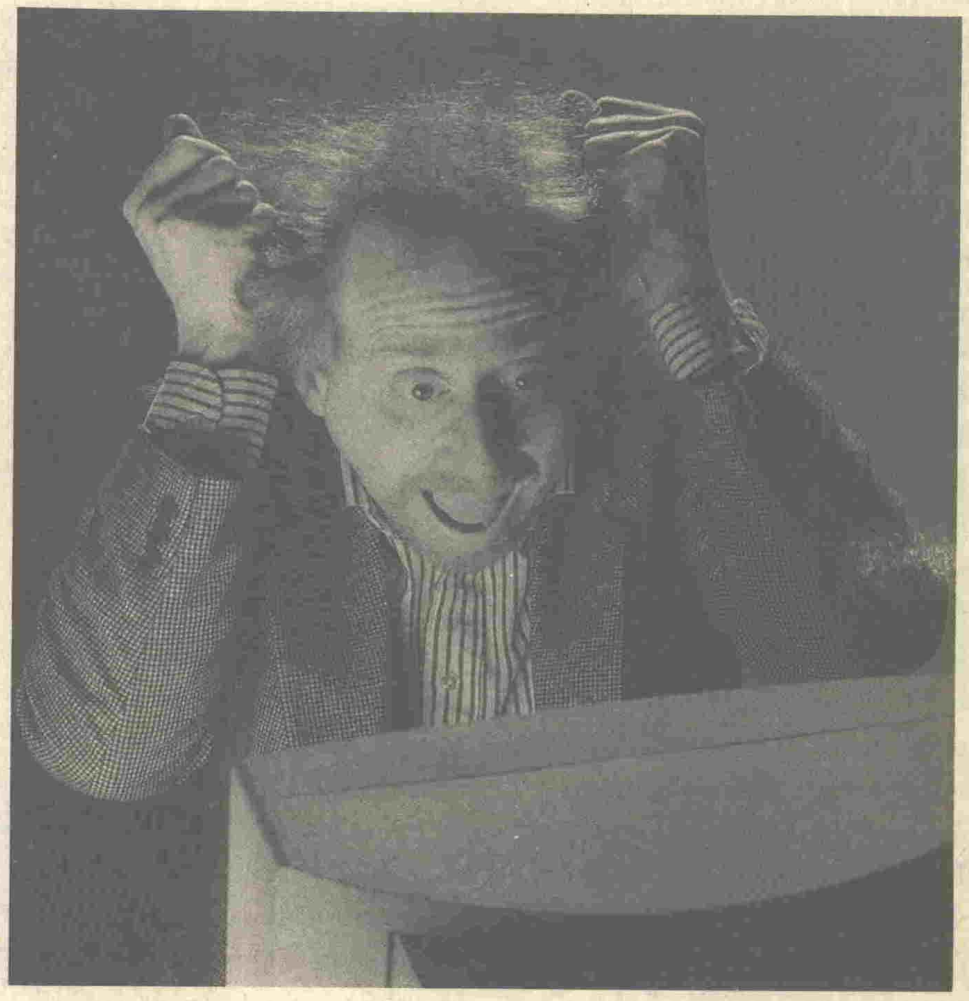
ซอฟต์แวร์
ประวิทย์ เรืองศาสตร์

พูดจาภาษา SQL ตอนที่ 1

แนะนำให้ผู้อ่านได้ทราบเรื่องราวของภาษา SQL ในขั้นพื้นฐาน จนถึงการนำไปประยุกต์ใช้งานได้จริง



หลังจากที่โปรแกรมดีเบสทรีพลัสประสบความสำเร็จในตลาดอย่างงดงามแล้ว ในเวลาต่อมาทางบริษัทแอตตัน-เทต ก็ได้ทำการเสริมกำลังรบอีกระลอก ด้วยการประกาศตัวดีเบสโฟร์ออกมา โดยถ้าหากมองรูปโฉมของดีเบสโฟร์ในตอนนั้นแล้ว ถ้าหากจะเปรียบเป็นหญิงสาว ก็เรียกว่า "สวยกินขาดบาดใจ" เลยทีเดียวล่ะ เพราะว่า (ในตอนนั้น) นอกจากดีเบสโฟร์จะได้เสริมสร้างเรื่องราวที่เป็นรากฐานเดิมจากโปรแกรมดีเบสทรีพลัสแล้ว ดีเบสโฟร์ก็ยังเป็นผู้กระตุ้นให้เกิดก้าวแห่งการพัฒนาในวงการฐานข้อมูลบนไมโครคอมพิวเตอร์อีกด้วย นั่นก็คือ การนำเอาภาษา SQL ผนวกเข้ามาในดีเบสโฟร์ด้วย ซึ่งตรงจุดนี้ก็ต้องขอยกความดีความชอบให้กับผู้ผลิตดีเบสโฟร์ไปได้เลย เพราะว่าเป็นผู้ทำให้ SQL เกิดขึ้นในโลกไมโครคอมพิวเตอร์ได้อย่างค่อนข้างเหมาะสม และสอดคล้องกับองค์ประกอบหลาย ๆ อย่างในขณะนั้น และจากนั้นต่อมาก็ได้เกิดการตื่นตัวกันพอสมควร ทั้งในแง่ของผู้ใช้ที่มีการตื่นตัวที่จะทำความรู้จักมักคุ้นกับ SQL กันมากขึ้น ส่วนในแง่ของผู้ผลิตซอฟต์แวร์ประเภท DBMP (DBMP : Database Management Package) ก็ได้มีการตื่นตัว (หรืออาจจะตื่นกลัวการเสียส่วนแบ่งในตลาดก็เป็นไปได้) เหมือนกัน เพราะว่าหลังจากนั้นไม่นานนักก็มีขบวนพาเหรดของซอฟต์แวร์ประเภท DBMP ที่ได้ปรับความสามารถและได้เพิ่มเติมส่วนที่เกี่ยวข้องกับภาษา SQL ออกเวอร์ชันใหม่สู่ตลาดตามออกมาเป็นทิวแถวเป็นต้นว่า PARADOX 3.5, RBASE:5000 Ver 3.0, FoxPro 2.0 และอื่น ๆ

ตกลงกันก่อน

โดยความตั้งใจของผู้เขียนเองแล้วเคยมีแนวความคิดว่าจะเขียนบทความเรื่องการใช้งาน SQL มาก่อนนี้ตั้งนาน

แล้ว แต่ว่าติดขัดอยู่ที่ในตอนนั้น ยังหาซอฟต์แวร์ที่จะใช้ประกอบได้ไม่ลงตัว เพราะว่าก่อนที่ฟอกซ์โปร 2.0 จะประกาศตัวออกมา ซอฟต์แวร์บนไมโครคอมพิวเตอร์ที่มีเชื้อ SQL อย่างเต็มตัวก็เห็นจะเป็นซอฟต์แวร์ที่ชื่อ Oracle แต่ด้วยเหตุว่า โอราเคิลนั้นเป็นซอฟต์แวร์ที่ค่อนข้างจะหิวหน่วยความจำพอสมควร เพราะว่าเครื่องพีซีที่จะใช้โอราเคิลจะต้องมีหน่วยความจำอย่างน้อย 2 เมกะไบต์ และถ้าจะให้ดี ในการใช้งานก็ควรจะเป็นเครื่องที่ใช้ซีพียูเบอร์ 80386 ขึ้นไป ซึ่งถ้าหากต้องให้ผู้อ่านเสียค่าใช้จ่ายส่วนนี้เพิ่ม เพื่อที่จะศึกษา SQL ก็ออกจะเป็นการลงทุนที่สูงไปสักหน่อย สำหรับตอนนี้ผู้เขียนได้เลือกซอฟต์แวร์ที่เราจะใช้ในการเรียนรู้ภาษา SQL ได้เรียบร้อยแล้ว (และคิดว่าเราค่อนข้างจะคุ้นเคย) โดยผู้เขียนจะใช้ฟอกซ์โปร 2.0 เป็นซอฟต์แวร์ที่จะใช้ประกอบในบทความชุดนี้ (นั่นหมายถึงว่าบทความนี้จะมีมากกว่าหนึ่งตอน) ซึ่งผู้เขียนมีจุดมุ่งหมายหลักคืออยากจะแนะนำให้ผู้อ่านได้ทราบเรื่องราวของภาษา SQL ในขั้นพื้นฐาน ไปจนถึงสามารถนำเอาภาษา SQL ไปประยุกต์ใช้งานได้จริง แต่ก่อนอื่นผู้เขียนต้องขอชี้แจงเอาไว้ก่อนว่า ภาษา SQL ที่มีในฟอกซ์โปร 2.0 เป็นเพียงส่วนย่อยของภาษา SQL มาตรฐาน ซึ่งผู้กำหนดมาตรฐานคือ ANSI (American National Standard Institute) และ ISO (International Standard Organization) แต่อย่างไรก็ตาม ผู้เขียนก็จะพยายามเสนออรรถรสของ SQL ให้ผู้อ่านได้ซึมทราบมากที่สุดเท่าที่จะทำได้ โดยผู้เขียนจะเน้นไปในแนวทางของภาคปฏิบัติมากกว่าทฤษฎี (แต่ยังไงก็ต้องมีทฤษฎีบ้าง จริงมั้ย) ครับผม

ฐานข้อมูลแบบสัมพันธ์ (Relational Database)

ฐานข้อมูลแบบสัมพันธ์ เป็นรูปแบบของโครงสร้างข้อมูลที่ถูกพัฒนาถัดมาจาก โครงสร้างข้อมูลแบบต้นไม้ (Hierachical Model) และโครงสร้างแบบเครือข่าย (Network Model) โดยแฟ้มข้อมูลแต่ละแฟ้มจะมีชื่อเฉพาะ และเราจะมองเห็นข้อมูลถูกจัดเก็บในรูปแบบของตารางสองมิติ ในตารางจะประกอบด้วยแถว (ROW) และคอลัมน์ (COLUMN) โดยที่ข้อมูลแต่ละแถวเรียกว่าเรคอร์ด และข้อมูลในแต่ละแถวก็จะประกอบด้วยข้อมูลที่เกี่ยวข้องกันตั้งแต่ 1 คอลัมน์ขึ้นไป โดยข้อมูลแต่ละคอลัมน์นี้เรียกว่า ฟิลด์ โดยมีชื่อฟิลด์ (Field Name) กำกับไว้ เพื่อใช้อ้างถึงรายละเอียดของข้อมูลภายในฟิลด์ นอกจากนี้แล้วโครงสร้างแบบนี้ยังสามารถเชื่อมโยงข้อมูลระหว่างตาราง (แฟ้มข้อมูล) โดยการใช้ฟิลด์ที่สมนัยกันเป็นตัวเชื่อมโยง ซึ่งก็จะทำให้เราสามารถนำเอาข้อมูลจากหลาย ๆ แฟ้มข้อมูลมาประมวลผลได้ในเวลาเดียวกัน (เพื่อให้ผู้อ่านทำความเข้าใจได้ง่ายยิ่งขึ้น ขอให้ดูตารางข้อมูลในรูปที่ 4 ประกอบไปด้วยนะครับ)

รูปที่ 1 ความหมายของฐานข้อมูลแบบสัมพันธ์ (อย่างคร่าว ๆ)

SQL คืออะไร?

อันที่จริงผู้เขียนไม่ได้มีเจตนาที่จะลากความเก่ามาเล่าซ้ำ เนื่องจากว่าเรื่องราวของ SQL ได้มีการเขียนถึงไว้มากพอสมควรแล้ว ในวารสารไมโครฯ ฉบับก่อน ๆ (ซึ่งท่านอาจจะไปค้นหาเพื่อดูประกอบก็ได้ครับ) แต่ผู้เขียนก็อยากจะเผื่อเหลือเผื่อขาดไว้สักนิดสำหรับผู้อ่านที่เพิ่งเริ่มต้น เพื่อให้ผู้อ่านได้มีสิทธิเท่าเทียมกันในการรับรู้ไงล่ะครับ ตรงนี้ผู้เขียนก็เลยขอถือโอกาสขยายความเรื่องราวความเป็นมาของ SQL อีกสักนิด ถ้าหากว่าท่านผู้ใดทราบอยู่แล้ว ก็สามารถข้ามเนื้อความส่วนนี้ไปได้เลยนะครับ

เอาล่ะ เริ่มกันเลยดีกว่านะครับ ตัวอักษร SQL ซึ่งย่อมาจาก คำว่า "Structure Query Langauge" และนิยมออกเสียงว่า "ซี-เควล" (Sequel) ซึ่งพอจะให้ความหมายอย่างกระชับได้ว่า SQL เป็นภาษาที่ใช้ (และอำนวยความสะดวก) สำหรับการจัดการและประมวลผลข้อมูล ที่วางตัวอยู่ในรูปแบบของฐานข้อมูลแบบสัมพันธ์ (Relational Database) โดยเฉพาะ (ผมได้อธิบายโครงสร้างอย่างคร่าว ๆ ของฐานข้อมูลแบบสัมพันธ์เอาไว้ในรูปที่ 1 แล้วครับ) และถ้าหากเราจะมองดู SQL กันให้ลึกลงไปอีก ก็จะพบว่าภาษา SQL มีส่วนประกอบที่สำคัญอยู่ 3 ส่วนด้วยกันคือ

1. **ส่วนกำหนดความหมายข้อมูล** (DDL : Data Definition Langauge หรือบางทีก็เรียกว่า Schema Definition Langauge) เป็นกลุ่มของคำสั่ง ที่ใช้ในการสร้างองค์ประกอบ

ต่าง ๆ เช่น ตารางข้อมูล (Base Tables), ดัชนีข้อมูล (Indexes), ตารางดูผลลัพธ์ (Views)

2. ส่วนการจัดการข้อมูล (DML : Data Manipulation Language) เป็นกลุ่มคำสั่งที่ใช้ในการเพิ่มเติม/แก้ไข/ลบข้อมูล และทำการตรวจสอบ/ค้นหาข้อมูลที่ถูกเก็บไว้ในตารางข้อมูล

3. ส่วนการควบคุมการเข้าถึงข้อมูล (DCL : Data Control Language) เป็นกลุ่มคำสั่งที่ใช้สำหรับการควบคุมและให้สิทธิ์ในการใช้ข้อมูล หรืออาจจะกล่าวได้อีกนัยหนึ่งว่าเป็นส่วนที่ใช้สำหรับการป้องกัน ไม่ให้มีการใช้ข้อมูลโดยไม่ได้รับอนุญาต (เพื่อความปลอดภัยของข้อมูลยังไงล่ะครับ)

โดยแรกเริ่มเดิมทีแล้วภาษา SQL ถูกออกแบบขึ้นโดย D.D. Chamberlin แห่งห้องปฏิบัติการวิจัยของไอบีเอ็ม ที่เมืองซานโจเซ มลรัฐแคลิฟอร์เนีย ในช่วงของทศวรรษ 1970 แต่ทว่าผู้ที่เป็นผู้นำของผลิตภัณฑ์ SQL ในเชิงพาณิชย์ กลับกลายเป็นบริษัท โอราเคิล คอร์เปอเรชัน จำกัด โดยเริ่มประกาศ SQL Database ครั้งแรกเมื่อ 12 ปีที่แล้ว และยังเป็นผู้พัฒนา SQL บนเครื่องพีซีเมื่อ 6 ปีที่แล้ว ซึ่งอาจจะกล่าวได้ว่าโอราเคิลเป็นซอฟต์แวร์ SQL ที่มีประสิทธิภาพน่าไว้วางใจมากที่สุดตัวหนึ่งเลยทีเดียว แต่ทว่าในตอนนี้สถานะของซอฟต์แวร์โอราเคิลค่อนข้างจะโดนซอฟต์แวร์ที่ชื่อว่า SQL Server ของบริษัทไมโครซอฟต์ทำคะแนนเบียดขึ้นมาหน่อย ๆ แล้ว (จากรายงานการทดสอบของวารสาร DBMS ประจำเดือนสิงหาคม พ.ศ. 2534)

ทำไมต้อง SQL?

เนื่องมาจากในปัจจุบันนี้ ข้อมูลข่าวสาร (Information) เป็นเรื่องที่ค่อนข้างจำเป็นสำหรับชีวิตประจำวันของเรา ๆ ท่าน ๆ (โดยเฉพาะคนที่อยู่ในวงการธุรกิจ) มากพอสมควรเลยทีเดียว ดังนั้นระบบฐานข้อมูลจึงได้เข้ามามีบทบาทมากขึ้น และเนื่องมาจากความต้องการข้อมูลข่าวสารนับวันก็ยิ่งจะทวีสูงขึ้น (ตามอัตราการแข่งขันทางธุรกิจ) ดังนั้นระบบฐานข้อมูลที่ใช้อยู่ก็นับวันที่จะขยายตัวและมีความซับซ้อนมากขึ้น (ผู้อ่านจะเห็นภาพได้ชัดเจนขึ้น ถ้าหากนึกถึงฐานข้อมูลที่มีข้อมูลกระจายอยู่ตามเครื่องคอมพิวเตอร์ที่ต่างขนาดกัน หรือใช้ระบบดำเนินการต่างกัน แต่ว่าข้อมูลข่าวสารที่ต้องการเป็นเรื่องค่อนข้างยืดหยุ่น (และต้องแข่งกับเวลา) ดังนั้นการพัฒนาระบบการจัดการฐานข้อมูล ให้สามารถรองรับความต้องการดังกล่าวจึงเป็นปัญหาที่ต้องขบคิดอย่างหนักของบรรดานักคอมพิวเตอร์ทั้งหลาย ทั้งนี้ก็เพราะว่าการพัฒนาระบบงานคอมพิวเตอร์เป็นงานที่ค่อนข้างละเอียดอ่อนและกินเวลามากพอสมควร ดังนั้นถ้าหากว่าในองค์กรใด มีการปรับเปลี่ยนความต้องการอยู่บ่อย ๆ ก็จะทำให้เหล่านักคอมพิวเตอร์รู้สึกปวดหัวตัวร้อน (หรืออย่างน้อยก็อาจจะอารมณ์เสีย) ไปตาม ๆ กัน เรื่องนี้ผมก็ต้องขออนุญาตเรียนให้ผู้อ่านทราบเป็นกรณีพิเศษ เพื่อให้ท่านเข้าใจลักษณะงานของนักคอมพิวเตอร์ เพราะว่าแต่ก่อนนั้น (แม้กระทั่งปัจจุบันก็ตาม) การพัฒนาระบบงานประยุกต์ของนักคอมพิวเตอร์ส่วนใหญ่ก็จะพึ่งพาอาศัยการเขียนโปรแกรมด้วย "ภาษายุคที่ 3" (3GL : Third Generation Langauge) ซึ่งภาษาที่นิยมใช้ก็ได้แก่ ภาษาเบสิก, ปาสคาล, ซี เป็นต้น และโดยธรรมชาติของภาษาเหล่านี้ซึ่งเป็นภาษาที่มีคำสั่งให้ใช้งานได้ทั่ว ๆ ไป แต่ไม่ได้เน้นหนักสำหรับการประยุกต์ใช้งานในเรื่องใดเรื่องหนึ่งโดยเฉพาะ ดังนั้นเมื่อนำเอาภาษาเหล่านี้มาสร้างระบบงานประยุกต์ก็ทำให้ในตอนเริ่มต้น นักคอมพิวเตอร์ต้องเริ่มต้นจากศูนย์ โดยเริ่มจากการศึกษาคำสั่งและไวยากรณ์ของภาษา

แล้วค่อย ๆ สร้างโปรแกรมย่อยรูทีนสะสมไว้ เพื่อใช้เป็นเครื่องมือในการสร้างระบบงานต่อไป ซึ่งถ้าหากจะเปรียบเทียบให้ท่านเห็นชัด ก็คงจะเหมือนกับการที่เราต้องการจะสร้างบ้านสักหลัง แต่เราต้องเริ่มต้นจากการเอาดินเหนียวมาปั้นเป็นก้อนอิฐ (สร้างรูทีน) แล้วเผาอิฐด้วยตัวเอง (ขัดเกลาและตรวจสอบรูทีน) แล้วค่อยเอาอิฐที่ได้มาก่อเป็นบ้าน ตามแปลน (ของระบบงาน) ที่เราออกแบบเอาไว้ ซึ่งท่านจะเห็นว่าเป็นเรื่องที่ไม่ง่ายนักเลยใช่มั๊ยครับ นอกจากนี้แล้วการใช้ 3GL ที่มีใช้กันอยู่ในปัจจุบันยังมีความยุ่งยากในเรื่องของการโยกย้ายระบบงานระหว่างระบบ (ถ้าหากมีการเปลี่ยนระบบคอมพิวเตอร์เกิดขึ้นในองค์กรของท่าน ซึ่งเป็นเรื่องน่าปวดหัวมากขึ้นไปอีก)

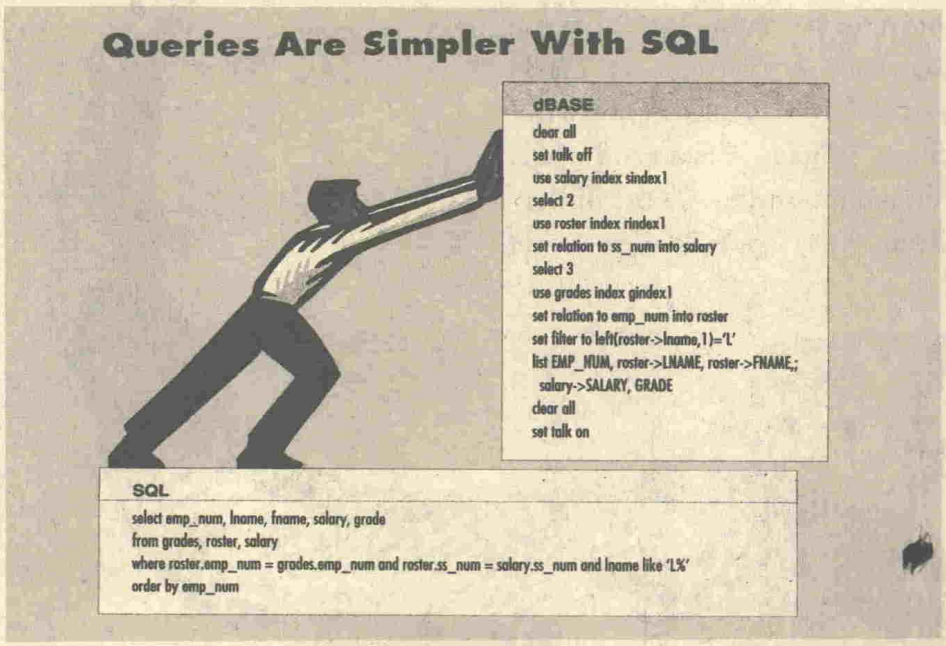
จากมูลเหตุที่ผมได้กล่าวไว้ข้างต้นนั้น ถ้าหากว่าท่านใช้ภาษา SQL ซึ่งถือได้ว่าเป็นภาษายุคที่ 4 (4GL : Fourth Generation Laguage) เข้าช่วยในการพัฒนาระบบงานประยุกต์ เพื่อจัดการกับระบบฐานข้อมูลในองค์กรของท่านแล้ว ปัญหาต่าง ๆ ก็จะลดน้อยลง (ไม่ใช่ว่าจะหมดไปนะครับ) ทั้งนี้ก็เพราะว่าจุดเด่นของภาษา SQL ซึ่งพอที่จะใช้เป็นเหตุผลสนับสนุน มีดังต่อไปนี้ครับ

มีมาตรฐาน

อาจจะเนื่องมาจากว่า SQL เป็นภาษาที่ทางบริษัทยักษ์ใหญ่อย่างไอบีเอ็มเป็นผู้พัฒนาขึ้นมาและยังได้ระบุให้ซอฟต์แวร์จัดการฐานข้อมูลของไอบีเอ็มเอง ในเครื่องคอมพิวเตอร์ทุกระดับ (เป็นเพียงความคิดเห็นส่วนตัวของผู้เขียนเท่านั้นนะครับ) ทำให้ภาษา SQL ได้รับความนิยมอย่างแพร่หลาย จึงทำให้มีภาษา SQL ก่อกำเนิดขึ้นมาจากผู้ผลิตซอฟต์แวร์หลาย ๆ ค่าย (ที่สำคัญก็ได้แก่ โอราเคิล, อินฟอร์มิกซ์, ไมโครซอฟต์, แอตตัน-เทต) จนกระทั่งเดือดร้อนถึงสำนักงาน ANSI ต้องทำการกำหนดมาตรฐานสำหรับภาษา SQL ขึ้น เพื่อใช้อย่างเป็นกลาง และต่อมาอีกไม่นานทางองค์กร ISO ก็ได้ยอมรับและได้กำหนดมาตรฐานสำหรับ SQL เช่นกัน ดังนั้นเราจึงสามารถใช้ภาษา SQL ได้บนเครื่องคอมพิวเตอร์รุ่นต่าง ๆ ด้วยวิธีการและขั้นตอนที่ (ค่อนข้าง) ใกล้เคียงกัน ซึ่งอาจจะถือเอาเป็นข้อสรุปได้ว่า ภาษา SQL ได้กลายเป็นมาตรฐานเคียงคู่กับระบบการจัดการฐานข้อมูลแบบสัมพันธ์ไปเสียแล้วครับ

ใช้งานก็ง่าย

ภาษา SQL เป็นภาษายุคที่สี่ ที่มีรูปแบบคำสั่งใกล้เคียงกับภาษาพูดของมนุษย์ ที่สำคัญคือ เมื่อเราใช้ภาษา SQL ในการเขียนโปรแกรม เราจะสามารถดำเนินการกับข้อมูลได้โดยไม่จำเป็นต้องยุ่งเกี่ยว หรือคำนึงว่า ข้อมูลในระดับกายภาพวางตัวอยู่อย่างไร แต่จุดสำคัญที่สุดคือ เมื่อเราใช้ภาษา SQL ในการเขียนโปรแกรม ก็คือเราจะเขียนโปรแกรมโดยใช้คำสั่งคล้ายกับการสั่งงานมนุษย์ (ซึ่งคิดเป็น) โดยจะบอกเพียงแต่ว่า "ต้องการผลลัพธ์อะไร?" โดยไม่ต้องบอกว่า "ทำอย่างไร?" (เพราะว่าเป็นหน้าที่ของ DBMS) ซึ่งจะทำให้เราเขียนโปรแกรมได้สั้น กระชับและสะดวกขึ้นกว่าการใช้ภาษายุคที่สาม ซึ่งเราต้องบอกกระบวนการอย่างค่อนข้างละเอียดและรัดกุมเลยทีเดียว และเพื่อเป็นการพิสูจน์ให้เห็นประสิทธิภาพของภาษา SQL ให้ชัดเจนยิ่งขึ้น ผมเลยขอเสนอตัวอย่างโปรแกรมที่เปรียบเทียบระหว่างโปรแกรมภาษา Xbase (ภาษาในตระกูลดีเบส) กับภาษา SQL ซึ่งได้แสดงไว้ในรูปที่ 2 โดยโปรแกรมทั้งสองจะให้ผลลัพธ์เหมือนกัน (แต่ว่าโปรแกรมภาษา SQL สั้นกว่าตั้งเยอะ)



รูปที่ 2 โปรแกรมเปรียบเทียบระหว่างภาษา Xbase กับภาษา SQL

ผสมผสานได้

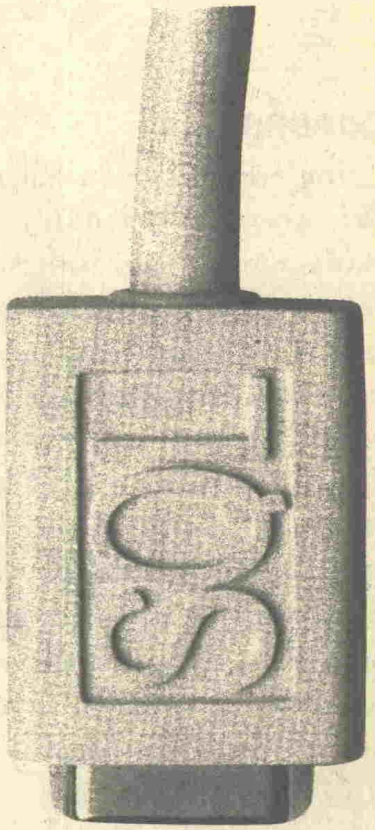
ถึงแม้ว่าตอนนี้คุณกำลังใช้ภาษาใด ๆ ในการเขียนโปรแกรมอยู่ก็ตาม ภาษา SQL ก็จะไม่มายื้อแย่งตำแหน่งกับภาษาที่คุณใช้อยู่เป็นประจำเป็นอันขาด (ซึ่งอาจจะทำให้คุณตัดสินใจลำบาก) ทั้งนี้ก็เพราะว่าภาษา SQL สามารถจะเขียนคำสั่งร่วมกับภาษาอื่น ๆ ได้ เช่น โคบอล, ซี โดยคุณสามารถใช้ภาษาที่คุณถนัดต่อไปได้ตามปกติ แล้วยังนำเอาภาษา SQL มาช่วยจัดการกับข้อมูลได้อย่างสะดวกสบายขึ้นอีกด้วย

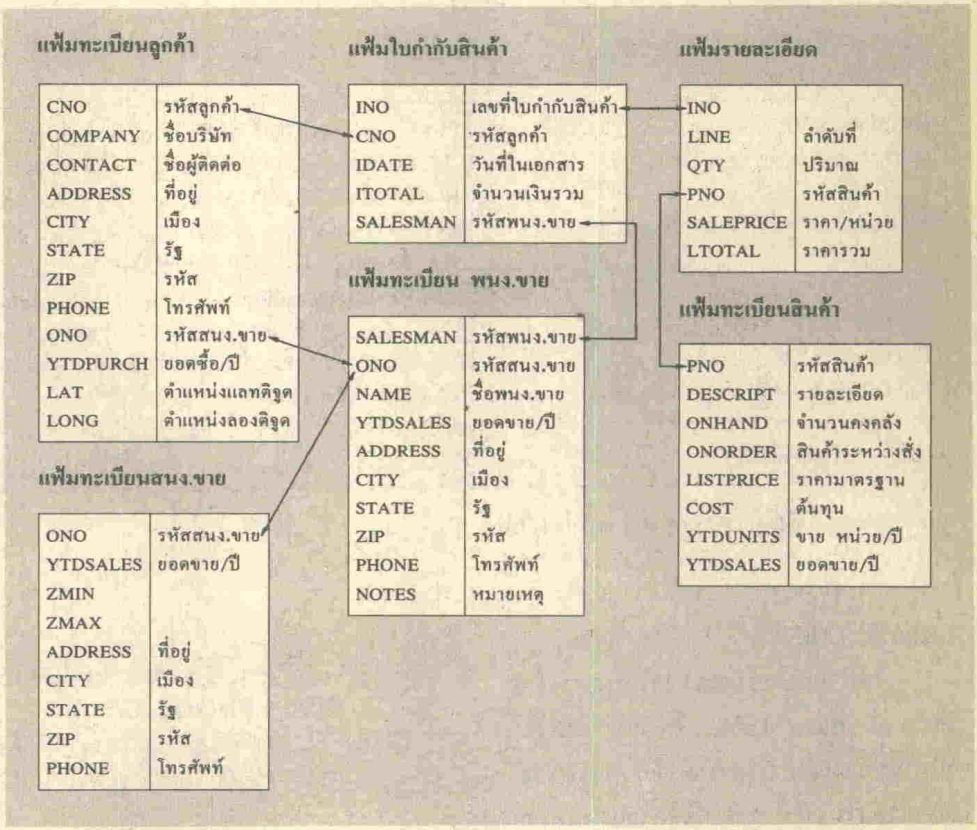
มุ่งสู่แนวใหม่

เป็นที่รู้ ๆ กันอยู่ว่า ในวงการคอมพิวเตอร์ย่อมมีพัฒนาการและการเปลี่ยนแปลงอยู่เสมอ ภาษา SQL ก็เช่นกัน จากจุดเริ่มต้นมาถึงวันนี้ก็มีอายุเกือบยี่สิบปีแล้ว ภาษา SQL ก็ได้มีการพัฒนาขึ้นมาโดยลำดับ โดยทาง ANSI จึงได้กำหนดมาตรฐาน SQL2 ขึ้นมาเพื่อแก้ไขข้อบกพร่องต่าง ๆ ของมาตรฐาน SQL ที่มีอยู่เดิมเป็นที่เรียบร้อยไปแล้ว นอกจากนี้แล้วทาง ANSI ยังกำลังทำการร่างมาตรฐาน SQL3 ซึ่งคาดว่าจะเรียบร้อยภายในปี ค.ศ. 1993/94 ซึ่งมีแนวโน้มว่าอาจจะมีส่วนของแนวคิดในเรื่องโปรแกรมเชิงวัตถุ (OOP : Object Oriented Programming) รวมอยู่ด้วย โดยในขณะนี้ก็มีผู้ผลิตซอฟต์แวร์สองราย คือ อินเกรสและฮิวเล็ตต์-แพคการ์ด กำลังขะมักเขม้นอยู่กับเรื่องที่ว่านี้ ดังนั้นจะเห็นว่าอนาคตของภาษา SQL จึงค่อนข้างสดใสและทันสมัยมากพอตัวทีเดียว สำหรับเรื่องนี้หากมีข่าวคราวคืบหน้าอย่างไร ผมก็จะหาโอกาสเอาเรื่องนี้มานำเสนออีกครั้งก็แล้วกันครับ

แจ้งให้ทราบ

ก่อนที่จะเริ่มผมต้องเรียนให้ทราบก่อนว่า เราจะใช้ตารางข้อมูลตามในรูปที่ 3 สำหรับเป็นตัวอย่างประกอบการอธิบายตลอดทั้งบทความ (ตารางทั้งหมดในรูปที่ 3 มีมากับฟอกซ์โปร 2.0 อยู่แล้ว โดยอยู่ในไดเรคทอรีย่อยที่ชื่อว่า TUTORIAL) สำหรับแฟ้มข้อมูลตัวอย่างทั้งหมดที่ใช้ในที่นี้ เป็นแฟ้มข้อมูลซึ่งออกแบบเพื่อใช้สำหรับระบบการขายสินค้า แต่ว่าอาจจะไม่ใช่ระบบที่สมบูรณ์แบบเท่าใดนัก ดังนั้นถ้าหากว่าผู้อ่านต้องการจะนำไปใช้ก็อาจจะต้องปรับแต่งอีกเล็กน้อย เพื่อให้สามารถใช้กับระบบงานของท่านได้อย่างเหมาะสม

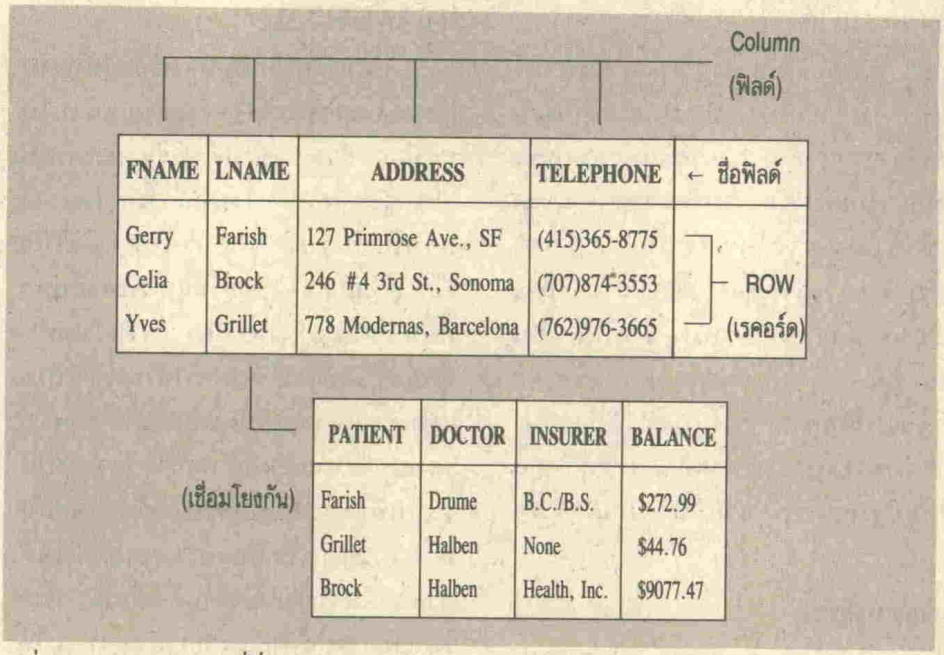




รูปที่ 3 ความสัมพันธ์ระหว่างแฟ้มข้อมูลในระบบขายสินค้า

เริ่มเลยดีกว่า

ผมได้เรียนให้ทราบตั้งแต่ตอนต้นแล้วว่า ภาษา SQL ในฟอกซ์โปร 2.0 เป็นเพียงส่วนย่อยของภาษา SQL มาตรฐาน ดังนั้นเรื่องที่เราจะเรียนรู้ภาษา SQL ในฟอกซ์โปรจึงง่ายกว่า เพราะว่าคำสั่งที่เกี่ยวข้องกับ SQL ในฟอกซ์โปรมีเพียง 4 คำสั่งคือ CREATE TABLE, CREATE QUERY, INSERT, SELECT, SET ANSI โดยในฟอกซ์-โปรจะไม่มีกลุ่มคำสั่งที่ทำหน้าที่ควบคุมการเข้าถึงข้อมูล (DCL) สำหรับรูปแบบเต็มของคำสั่ง SQL ทั้งหมด ผมได้แสดงเอาไว้ในรูปที่ 5 เพื่อใช้อ้างอิงตลอดทั้งบทความ โดยแต่ละคำสั่งมีหน้าที่ดังต่อไปนี้



รูปที่ 4 ตัวอย่างตารางข้อมูลที่เชื่อมโยงกัน

CREATE TABLE ! DBF <dbf_name> (<fname1> <type> [(<precision> [, <scale>]) [, < fname 2>...]]) ! FROM ARRAY <array>	CREATE QUERY [<file> ! ?] MODIFY QUERY [<file> ! ?] [NOWAIT]
INSERT INTO <dbf_name> [(<fname1] [, <fname2> [, ...]])] VALUES (<expr1> [, <expr2> [, ...]]) INSERT INTO <dbf_name> FROM ARRAY <array> FROM MEMVAR	SELECT [ALL ! DISTINCT] [<alias>.]<select_item>[AS<column_name>] [, [<alias>.]<select_item> [AS <column_name>] ...] FROM <database> [<local_alias>] [, <database> [<local_alias>] ...] [[INTO<destination>] ! [TO FILE <file> [ADDITIVE] ! TO PRINTER] [NOCONSOLE] [PLAIN] [NOWAIT] [WHERE <joincondition> [AND <joincondition> ...] [AND ! OR>filtercondition> [AND ! OR <filtercondition>...]]] [GROUP BY <groupcolumn> [, <groupcolumn> ...]] [HAVING <filtercondition>] [UNION [ALL] <SELECT command>] [ORDER BY <order_item> [ASC ! DESC][, <order_item>
SET ANSI NO ! OFF	

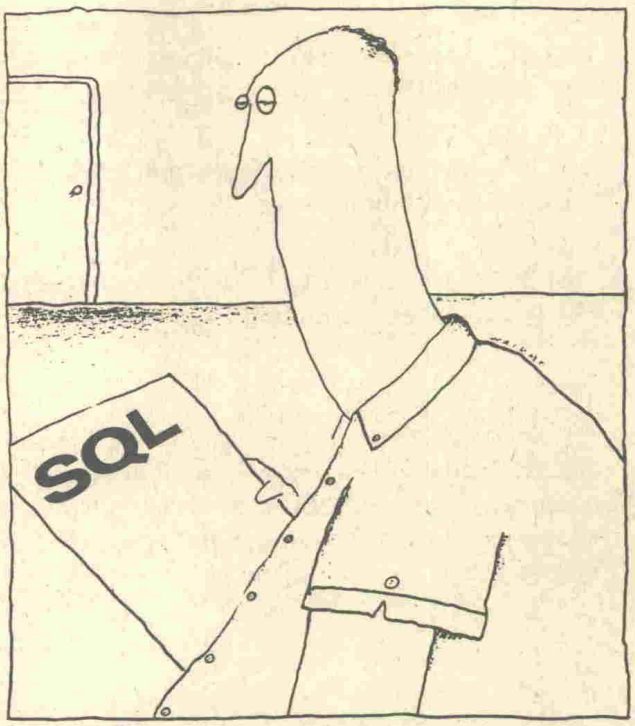
รูปที่ 5 รูปแบบคำสั่งที่เกี่ยวข้องกับ SQL ของฟอกซ์โปร

CREATE TABLE

ใช้สำหรับสร้างตารางข้อมูล (หรือเรียกว่า Base Table ในภาษา SQL ยกตัวอย่างเช่น ถ้าหากเราต้องการสร้างตารางดังในรูปที่ 4 ก็ให้ใช้คำสั่งดังแสดงไว้ในรูปที่ 6

```
CREATE TABLE person; (FNAME C(20), LNAME C(20), ADDRESS
    C(20), PHONE C(20))
CREATE TABLE patient; (PATIENT C(20), DOCTOR C(20),
    INSURER C(20), BALANCE N(10,2))
```

รูปที่ 6



CREATE QUERY

ใช้สำหรับสร้างแฟ้มที่กำหนดเงื่อนไข เพื่อดึงข้อมูลที่ต้องการจากฐานข้อมูล โดยเมื่อท่านใช้คำสั่งนี้ ฟอกซ์โปรก็จะแสดง RQBE Window เพื่อให้ท่านกำหนดเงื่อนไขต่าง ๆ เพื่อสอบถามข้อมูลตามที่ต้องการ ได้ทั้งในลักษณะการโต้ตอบแบบทันทีทันใด หรือถ้าหากท่านกำหนดเงื่อนไขในการดึงข้อมูลแล้วและต้องการจัดเก็บเอาไว้ ก็ให้ทำการเซฟเงื่อนไขเหล่านั้นก็จะถูกเก็บไว้ในแฟ้มที่มีส่วนขยายเป็น .QPR ซึ่งสามารถนำมาใช้งานได้อีกทุกเมื่อที่ต้องการ (รายละเอียดเรื่องการสร้าง Query จะกล่าวถึงอีกครั้งในตอนต่อไป)

INSERT

ใช้ในการเพิ่มเรคอร์ดเข้าในตารางข้อมูล ยกตัวอย่างเช่น ถ้าหากเราต้องการเพิ่มข้อมูลเข้าในตารางรูปที่ 4 ก็สามารถทำได้ดังแสดงไว้ในรูปที่ 7

SELECT

ใช้ในการดึงข้อมูลที่ต้องการจากฐานข้อมูล คำสั่งนี้ถือได้ว่าเป็นแกนสำคัญของ SQL เลยก็ว่าได้ ในฟอกซ์โปรท่านสามารถใช้คำสั่ง SELECT ได้หลายทางคือ ใช้ด้วยการพิมพ์คำสั่งที่คอมมานด์วินโดว์, ใช้คำสั่งร่วมกับโปรแกรมที่เขียนด้วยภาษาฟอกซ์โปร, หรือใช้งานผ่าน RQBE Window ก็ได้

SET ANSI

ใช้เพื่อระบุวิธีการเปรียบเทียบข้อมูลที่เป็นข้อความในคำสั่ง SQL ถ้าหากใช้คำสั่ง SET ANSI ON การเปรียบก็จะเป็นแบบตัวต่อตัว โดยถือความยาวของข้อความที่ยาวกว่าเป็นหลัก (ให้ผลคล้าย ๆ กับคำสั่ง SET EXACT ON : OFF) แต่ถ้าหาก SET ANSI OFF ก็จะทำการเปรียบเทียบเฉพาะความยาวของข้อความที่สั้นกว่า

```
INSERT INTO persons (FNAME, LNAME, ADDRESS, PHONE),
VALUES ("Prawit", "Ruengsart", HappySoft Co., Ltd.", "(02)
    392-1978")
```

รูปที่ 7

```
SET PATH TO C:\FOXPRO2\TUTORIAL\ <Enter>
SELECT A        <Enter>
USE customer    <Enter>
SELECT B        <Enter>
USE offices     <Enter>
SELECT C        <Enter>
USE invoices    <Enter>
```

รูปที่ 8

ทดลองใช้

เมื่อเราได้ทราบถึงคำสั่งต่าง ๆ ของ SQL ที่มีในฟอกซ์โปรไป (อย่างคร่าว ๆ) เรียบร้อยแล้ว ตอนนี้ก็ลองมาใช้คำสั่ง SELECT เพื่อทำการดึงข้อมูลจากแฟ้มข้อมูลตัวอย่างกันดูสักหน่อย โดยผู้เขียนจะมีโจทย์ฝึกหัดให้เป็นข้อ ๆ พร้อมกับตัวอย่างของคำสั่ง SQL ที่ใช้เพื่อดึงข้อมูลให้ได้ผลลัพธ์ตามโจทย์ ซึ่งถ้าหากผู้อ่านอยากจะ

```
– SELECT company FROM customer WHERE
  company LIKE "% Computer %"
หรือ
– SELECT company FROM customer WHERE
  "Computer" $ company
หรือ
– SELECT company FROM customer WHERE
  AT ("Computer", company) > 0
```

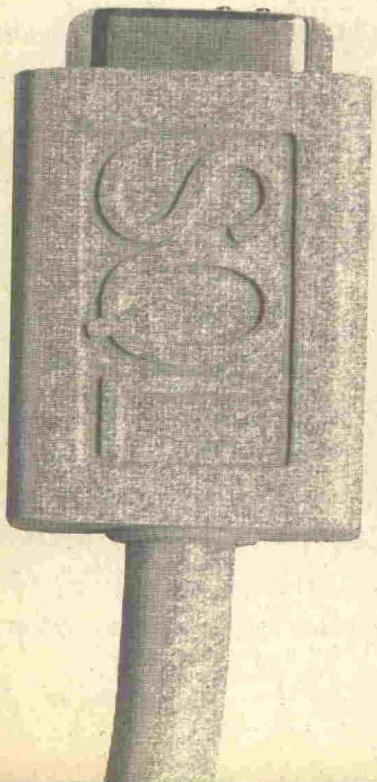
รูปที่ 9

```
– SELECT customer.company, customer.city
  customer.state; FORM customer, offices WHERE
  customer.city = offices.city; AND customer.state =
  offices.state
```

รูปที่ 10

```
– SELECT company FROM customer WHERE;
  cno NOT IN (SELECT cno FROM invoices)
หรือ
– SELECT company FROM customer WHERE
  NOT EXISTS; (SELECT * FROM invoices WHERE
  invoices.cno = customer.cno)
```

รูปที่ 11



ประลองฝีมือกับภาษา SQL ก็ให้ทดลองเขียนโปรแกรมด้วยคำสั่งปกติของฟอกซ์-โปรเพื่อให้ได้ผลลัพธ์อย่างเดียวกัน ก็จะทำให้เห็นชัดขึ้นว่า คำสั่ง SELECT มีประสิทธิภาพเพียงใด สำหรับโจทย์ตัวอย่างที่ผู้เขียนเลือกมา มีด้วยกัน 3 ข้อดังต่อไปนี้ครับ แต่ก่อนที่จะดำเนินการใด ๆ ให้ผู้อ่านทำการเปิดแฟ้มข้อมูลโดยการพิมพ์คำสั่งที่คอมมานด์วินโดว์ ดังแสดงไว้ในรูปที่ 8

จากนั้นให้ท่านทดลองใช้คำสั่งตามคำเฉลยในโจทย์แต่ละข้อ โดยพิมพ์คำสั่งที่คอมมานด์วินโดว์ได้เลย

โจทย์ข้อ 1.

ให้แสดงชื่อบริษัทของลูกค้า ที่มีชื่อบริษัทประกอบด้วยคำว่า "Computer" สามารถใช้คำสั่ง SELECT เพื่อดำเนินการได้ ดังแสดงไว้ในรูปที่ 9

โจทย์ข้อ 2.

ให้แสดง ชื่อบริษัท, เมือง, รัฐ ของลูกค้า ที่อยู่ในเมือง, รัฐ เดียวกันกับสำนักงานขาย สามารถใช้คำสั่ง SELECT เพื่อดำเนินการได้ ดังแสดงไว้ในรูปที่ 10

โจทย์ข้อ 3.

ให้แสดงชื่อบริษัทที่ไม่มีการสั่งซื้อสินค้า ซึ่งใช้คำสั่ง SELECT เพื่อดำเนินการได้ ดังแสดงไว้ในรูปที่ 11

ตอนต่อไป

สำหรับในบทความตอนนี้ ผู้เขียนอยากจะเรียกว่าเป็นการชิมรสภาษา SQL ในเบื้องต้นเท่านั้น ดังนั้นจึงยังไม่ได้แสดงถึงประโยชน์ในแง่มุมปฏิบัติของภาษา SQL ให้กระจ่างชัดเท่าใดนัก ซึ่งดู ๆ แล้วอาจจะเหมือนว่าเป็นการเชียร์ภาษา SQL เสียเกินกว่าเหตุ แต่ความจริงแล้วไม่ใช่อย่างนั้น เพราะว่าในเรื่องของการประยุกต์ใช้ผู้เขียนก็ได้เตรียมเอาไว้แล้วเช่นกัน ซึ่งจะพบได้ในบทความตอนต่อไป อ้อ! สำหรับบทความตอนหน้านี้ ก็จะเป็นเรื่องของการใช้คำสั่ง SELECT ที่จะอธิบายอย่างละเอียดทุกแง่มุม และยังมีของแถมให้ผู้อ่านอีกด้วยครับ คือผู้เขียนได้ไปเสาะหาส่วนที่เป็นข้อบกพร่องของ SQL (ที่ฝรั่งเขาวิจารณ์เอาไว้) เอามาแจกแจงให้ทราบกัน ซึ่งก็ขอยกยอดเอาไว้ตอนหน้าด้วย แล้วจะรวบรวมเอามาเรียนให้ทราบกันแบบรวบยอดไปเลย แล้วค่อยเจอกันใหม่ครับ